BusinessLink Software Support

Pocket Strategi Reference

Version V2R1

This manual applies to Strategi version V2R1 and later and was last revised in August, 2005.

A Note to Readers

The latest versions of this document and other Technical Support Bulletins can be downloaded from ADVANCED BusinessLink Corp.'s Support Website, http://support.businesslink.com.

You may print this in duplex format using Adobe's Acrobat Reader, which is available for download from http://www.adobe.com/products/acrobat/readstep.html. This document is designed for two-sided printing, which will result in occasional blank pages if viewing the document online.

With some installs of Adobe Acrobat, your printer may not resolve the characters correctly, and once printed, all characters will appear as rectangles or as symbols. If this happens, you will need to select "Print as image" from the Acrobat print dialogue. This will cause the print to occur correctly.

If you have any questions, comments or suggestions, please feel free to contact BusinessLink Software Support via email at support@businesslink.com.

Sincerely,

BusinessLink Software Support

Table of Contents

Introduction

Pocket Strategi is a tool for building applications and is a component of Strategi that operates on a device with a fully capable Java Virtual Machine (JVM) to take advantage of platform independence. This allows development, deployment and secure two-way communication between a Strategi client and host.

Pocket Strategi is also an HSM-enabled web server with supporting infrastructure to allow communication between host Strategi and Pocket Strategi Client via TCP/IP using internal Strategi protocols.

Pocket Strategi is ideally suited for solving business problems involving wireless web requirements without having to know how to program wireless devices. This allows application programmers to focus on business logic instead of complicated communications programming.

Pocket Strategi is not required to be connected to the same network as host Strategi, and as such, a connection between host Strategi and Pocket Strategi cannot be presumed. Before host Strategi will accept a Pocket Strategi Client connection, it must receive valid authentication information from Pocket Strategi Client. When starting Pocket Strategi, a connection to host Strategi does not automatically occur. A request must be made to connect to host Strategi; however, Pocket Strategi makes issuing the connection request as simple as possible using HSM.

Pocket Strategi is designed to support three basic categories of applications: always-connected, sometimes-connected and never-connected to the host.

Always-connected is designed for situations where a connection to the host is always expected.

Sometimes-connected is the most popular because user experience is consistent and independent of current connectivity status. If designed properly, the Pocket Strategi user may be oblivious to current connectivity status and not concern themselves with it until specific times, all of which is taken into account at design time.

Never-connected applications do not require connections to the host for application support. All information, data, images, files, etc., necessary to support the application reside on the client. One example of a never connected application is a troubleshooting guide that displays Portable Document Format (PDF) diagrams of schematics. Field engineers with hand-held devices running Pocket Strategi are able to access the troubleshooting guide application any place they can carry the hand-held device.

An applications developer building applications with Pocket Strategi should have a working knowledge of the following technical areas:

- Hyper Text Markup Language (HTML)
- Java basics (enough to meet business requirements)
- Strategi's HSM

Online technical resources are available from ADVANCED BusinessLink's website http://support.businesslink.com for product support, documentation and downloads.

Getting Started

The RESOURCES website of your host Strategi installation contains Pocket Strategi Client. Using standard Strategi website security, strict control limiting access to downloading Pocket Strategi Client is set by default. The Strategi administrator must provide *READ access to the PSTRATEGI website zone for each Strategi user or group authorized to download Pocket Strategi Client.

Pocket Strategi New User Configuration details are located in the Administration section of this document.

Client Download

In order to download Pocket Strategi Client, you must be authorized for Pocket Strategi access. Contact your Strategi Administrator for authorization and access details.

To download Pocket Strategi Client, access the Pocket Strategi New User Install homepage on the RESOURCES website using an Internet browser. Different download file types are available based upon target operating system platform, so click on the hyperlink that best describes the target platform for Pocket Strategi Client and begin the download. Additional links to installation instructions are found on the homepage for viewing.

http://yourIPAddress/resources/pstrategi/homepage.htm

Installation

Once Pocket Strategi Client is downloaded onto the target client device, follow the installation instructions provided on the instructions.html page located in the PSTRATEGI website zone of the RESOURCES website.

http://yourIPAddress/resources/pstrategi/instructions.html

The installation process allows for installing Pocket Strategi into the directory location of the user's choice. The Pocket Strategi components have a distinct and pre-set directory structure.

Component Overview

The Pocket Strategi Client root directory contains the core application executables and required sub-directories for successful operation. The following is a brief description of each major component found in the Pocket Strategi root and sub-directories.

pStrategi.jar

The *pStrategi.jar* file contains the startup classes for Pocket Strategi.

pStrategiClient.jar

The *pStrategiClient.jar* contains the majority of Pocket Strategi functionality classes, such as the web server and connection engine.

pStrategiClient-nnn.log

The *pStrategiClient-nnn.log* files are generated once the Pocket Strategi client is launched. The log files contain a log of Pocket Strategi client's activities, such as startup, initialization and connection information.

pStrategiUpdate.jar

The *pStrategiUpdate.jar* contains the classes used to update the Pocket Strategi client.

application

The *application* directory contains all database files intended for MiniDBMS interaction.

database

The *database* directory contains all Pocket Strategi configuration files. The directory contains the following database files upon initial install:

cv.dbf – Configuration Values File
cv.dfn – Configuration Values File Definition
eh.dbf – Event Handler Configuration File
eh.dfn – Event Handler Configuration File Definition
fh.dbf – From Host Tracking File (used internally by Pocket Strategi)
fh.dfn – From Host Tracking File Definition (used internally by Pocket Strategi)
hm.dbf – HSM Server Configuration File
hm.dfn – HSM Server Configuration File Definition
tg.dbf – Transfer Group Configuration File
tg.dfn – Transfer Group Configuration File Definition
th.dbf – To Host Tracking File (used internally by Pocket Strategi)
th.dfn – To Host Tracking File Definition (used internally by Pocket Strategi)

The following tables describe user-modifiable Pocket Strategi database files, listing the column name followed by the description.

Configuration Values File

Column	**Description**
CVKWD	Keyword
CVGRP	Keyword Group
CVDSC	Keyword Description
CVVAL	Keyword Value
CVUSRM	Keyword User Modifiable Flag

Event Handler Configuration File

Column	**Description**
EHCODE	Event Handler Code
EHDESC	Event Handler Description
EHCLAS	Event Handler Class Name
EHCLSP	Event Handler Class Path Relative to Pocket Strategi Root

HSM Server Configuration File

Column	**Description**
HMSVR	Server Name (Maximum length of 8 characters)
HMAUTO	Auto Start Flag (0=no; 1=yes)
HMINST	Number of Server Instances (S=single instance)
HMDESC	Server Description
HMCLAS	Class Name (*HOST if server defined on host Strategi)
HMCLSP	Class Path Relative to Pocket Strategi Root

Note that inserting blank lines between HSM Server Definitions will result in errors when Pocket Strategi attempts to launch a non-existent HSM Server.

Transfer Group Configuration File

Column	Description
TGGRP	Transfer Group
TGDESC	Transfer Group Description
TGTRAC	Transfer Group Tracking Number (automatically updated by Pocket Strategi)
TGEHRW	Event Handler Run When (0=online, immediately after transfer completes; 1=offline, following disconnect)
TGEHRH	Event Handler Run How (should be set to 0)

The cv.dbf file contains Pocket Strategi core configuration files and should be handled carefully. The following tables detail intended usage of each keyword contained in the file.

Keyword:	ACCESSNAME
Purpose:	Access name used to authenticate the Pocket Strategi user to the host system.
Default Value:	Blank upon install. Populated by the user upon first connection to host attempt.
Valid Value(s):	No restrictions
User Modifiable:	No
Default Value:	0
Valid Value(s):	0, 1

Keyword:	CHALLENGEKEY1
Purpose:	Part 1 of the challenge key response to the host system. Used during Strategi user authentication.
Default Value:	Blank upon install. Populated by Pocket Strategi upon first successful connection to host.
Valid Value(s):	No restrictions
User Modifiable:	No
Default Value:	0
Valid Value(s):	0, 1

Keyword:	CHALLENGEKEY2
Purpose:	Part 2 of the challenge key response to the host system. Used during Strategi user authentication.
Default Value:	Blank upon install. Populated by Pocket Strategi upon first successful connection to host.
Valid Value(s):	No restrictions
User Modifiable:	No
Default Value:	0
Valid Value(s):	0, 1

Keyword:	USERNUMBER
Purpose:	Strategi user number. Uniquely identifies the Strategi user to the host system.
Default Value:	Blank upon install. Populated by Pocket Strategi upon first successful connection to host.
Valid Value(s):	No restrictions
User Modifiable:	No
Default Value:	0
Valid Value(s):	0, 1

Keyword:	HOSTNETWORKNAME
Purpose:	Host Network Name or IP address Pocket Strategi must use to connect to the host system.
Default Value:	Blank upon install. Populated by Pocket Strategi upon first successful connection to host.
Valid Value(s):	No restrictions. Value used follows DNS naming conventions (e.g., www.businesslink.com or 192.1.1.1)
User Modifiable:	Yes
Default Value:	1
Valid Value(s):	0, 1

Keyword:	HOSTNETWORKPORT
Purpose:	HTTP host-port used for communicating with Pocket Strategi.
Default Value:	43840
Valid Value(s):	Any valid port number
User Modifiable:	Yes
Default Value:	1
Valid Value(s):	0, 1

Keyword:	LOCALNETWORKNAME
Purpose:	(Not Currently Used)
Default Value:	(Not Currently Used)
Valid Value(s):	(Not Currently Used)
User Modifiable:	(Not Currently Used)
Default Value:	(Not Currently Used)
Valid Value(s):	(Not Currently Used)

Keyword:	ABORTONLAUNCHERROR
Purpose:	Abort (shut down) Pocket Strategi if an error occurs launching the control program and/or control class (see LAUNCHCONTROLCLASS and LAUNCHCONTROLPROGRAM)
Default Value:	1
Valid Value(s):	0, 1 (0=log exception but continue; 1=abort Pocket Strategi)
User Modifiable:	Yes
Default Value:	1
Valid Value(s):	0,1

Keyword:	LAUNCHCONTROLCLASS
Purpose:	Launch a separate control class in addition to Pocket Strategi.
Default Value:	*NONE
Valid Value(s):	Any valid class name (e.g., MyClass). Maximum length of 128 characters, including parameters; command parameter style; double-quoted parameters allowed for blanks or special characters; do not include path to class; do not include class name extension (e.g., .class).
User Modifiable:	Yes
Default Value:	1
Valid Value(s):	0, 1

Keyword:	LAUNCHCONTROLPROGRAM
Purpose:	Launch a separate control program in addition to Pocket Strategi.
Default Value:	C:\Program Files\Internet Explorer\iexplore.exe http://127.0.0.1/homepage.htm

Valid Value(s):	*NONE, Any valid program name. Maximum length of 128 characters, including parameters; command parameter style; double-quoted parameters allowed for blanks or special characters.
User Modifiable:	Yes
Default Value:	1
Valid Value(s):	0, 1

eventhandlers

The *eventhandlers* directory contains all Event Handler classes. The directory is empty upon initial install.

hsmservers

The *hsmservers* directory contains all HSM Server classes. The directory is empty upon initial install.

UninstallerData

The *UninstallerData* directory contains the components necessary to un-install Pocket Strategi.

user

The *user* directory contains user-specific data intended for Pocket Strategi Client use only. The directory is not empty upon initial install.

website

The *website* directory and all its sub-directories contain the live website-related files. Currently only one website, default, is available to the Pocket Strategi web server for web page delivery. The directory is populated with additional sub-directories upon install.

The root directory for all live website files within the 'website' directory is website/default/live. There are three sub-directories in 'live' as part of the initial Pocket Strategi Client installation: *FirstTime*, *Messages* and *TestPages*.

FirstTime
The *FirstTime* sub-directory contains objects used to initiate a connection to Strategi host for user identity establishment and personalized application components download.

Messages
The *Messages* sub-directory is available for using customized error pages.

TestPages
The *TestPages* sub-directory contains objects used to test basic Pocket Strategi functionality, such as Startup, Shutdown, Connect-to-Host, To- and From-Host File Transfer, etc.

First Time Connection

After Pocket Strategi Client is installed on the Java-enabled device, start Pocket Strategi by executing the platform-specific executable or pStrategi.jar file located in the Pocket Strategi root directory.

Pocket Strategi will launch and load the local web server listening on TCP/IP address 127.0.0.1. By default, Pocket Strategi is configured to listen on TCP/IP address 127.0.0.1

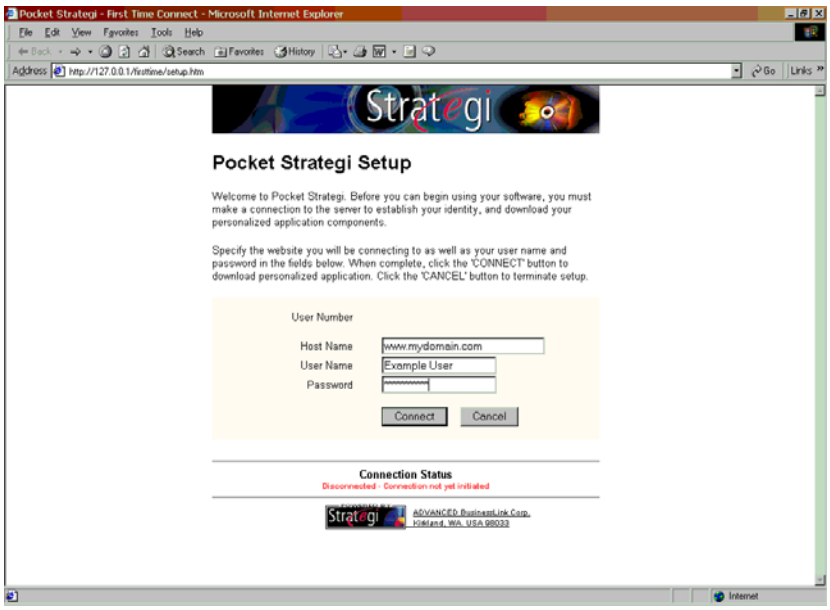
and launch Internet Explorer as its Launch Control Program. These values are configurable on the client in the cv.dbf file.

Upon successful launch, Pocket Strategi will serve "firsttime/setup.htm" to allow the user to make their first time connection to the host for identity establishment and personalized application components download. There are three required inputs to make the first time connection to host Strategi: *Host Name*, *User Name* and *Password*.

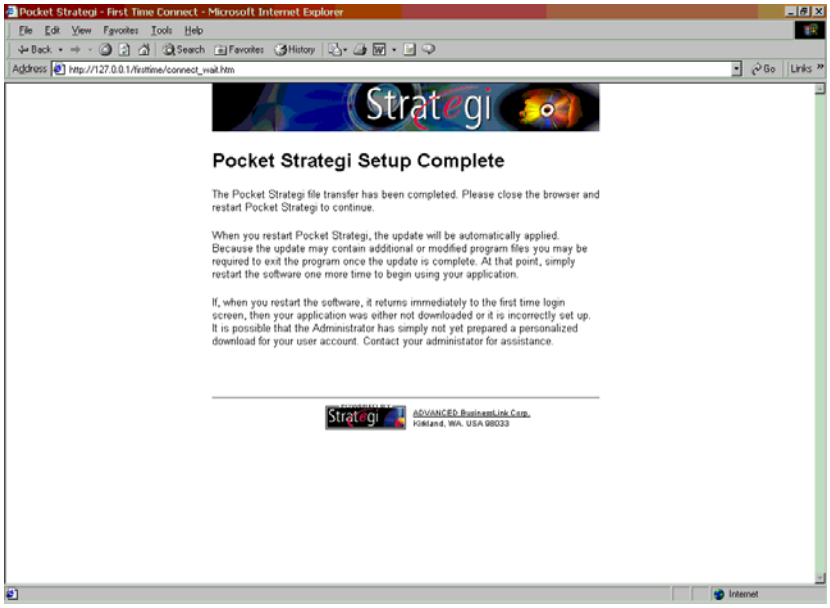
Host Name identifies the Domain Name Service (DNS) or TCP/IP address of the Strategi host. Using the DNS name instead of TCP/IP address is the preferred method.

User Name uniquely establishes the Pocket Strategi user's identity to host Strategi. The User Name must already exist on the host prior to initiating the connection, and is done so by executing the xxxSGIUSR host commands.

Password is the passphrase of the Strategi user (User Name) attempting identity establishment.

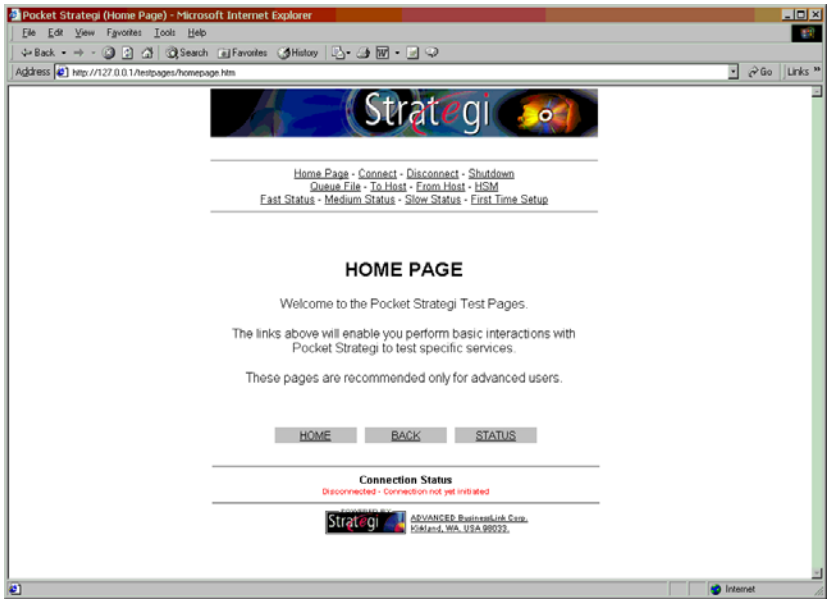


Upon successful connection to the host, the "Pocket Strategi Setup Complete" page will display. Instructions located on the page will direct the user.



Test Pages

Pocket Strategi's TestPages website sub-directory contains resources to conduct Pocket Strategi services testing. The following actions are included in TestPages: Connect, Disconnect, Shutdown, Queue File, To Host, From Host, HSM, and Status.



Home Page

Home Page (homepage.htm) is the logical starting point for navigating within Test Pages. The default URL value for a Pocket Strategi website is "homepage."

Connect

Connect demonstrates how to initiate a Pocket Strategi connection to host Strategi by issuing the CONNECT opcode. The initial page gets previous connection information via the GETCONNECT opcode, such as Host Name, User Name, and Password. If no prior host connection was made, the fields are blank. Otherwise, simply click on the "Connect" button to connect using the previously entered Host Name and User Name.

A Password entry is not required on every connection request because a long challenge key is stored locally in the cv.dbf file. The challenge key is changed for every connection, thus providing greater security than simply entering a password. Also, for many types of Pocket Strategi-based applications, requiring the user to enter their connection details each time may be impractical, especially in "sometimes connected" scenarios.

The user may elect to key in their password, and if so, must exactly match their host Strategi password. If incorrect, they will not obtain a host connection. Client application design will dictate which pieces of connection information are required, if any, other than the first time connection.

Disconnect

Disconnect causes Pocket Strategi to disconnect from the host by issuing the DISCONNECT opcode.

Shutdown

Shutdown causes Pocket Strategi to end its processes and close by issuing the SHUTDOWN opcode.

Queue File

Queue File prepares a file for transfer to host Strategi by issuing the SENDFILE opcode. In order to transfer (send) a file from Pocket Strategi to host Strategi, the file must first be queued. Any number of files can be queued, they are simply grouped according to Transfer Group.

When a file is queued, Pocket Strategi will store certain information about the file so it can be programmatically handled on the host by an Event Handler. The type of information stored is User Number, File Name, Transfer Group, Event Code, and Properties.

User Number is the Strategi user number assigned to the Pocket Strategi user during creation.

File Name is the name of the queued file for transfer.

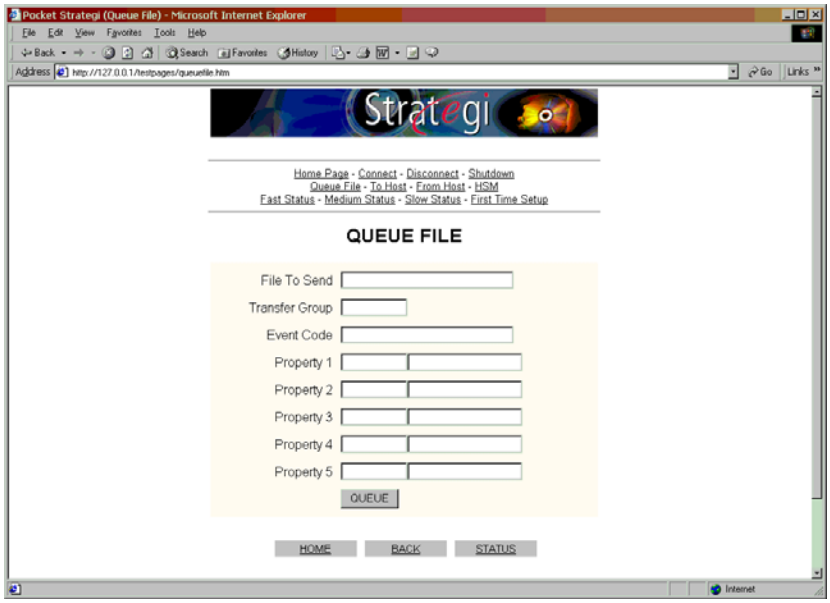
Transfer Group is a logical collection of related files that allow you to manage what gets moved to and from the host. Transfer Groups deal with application-to-application data delivery, and are designed purely for the client's benefit. The operating environment of the client may be limited in connection speed, bandwidth, memory, and other device constraints. By using Transfer Groups it allows the application designer to leverage file transfer efficiency.

When queuing a file for transfer, the Transfer Group must be defined in the tg.dbf file.

The Transfer Group value is an available standard property to the Event handler receiving the transferred file.

Event Code is used by Strategi's internal event clients to determine which handler to invoke for the event.

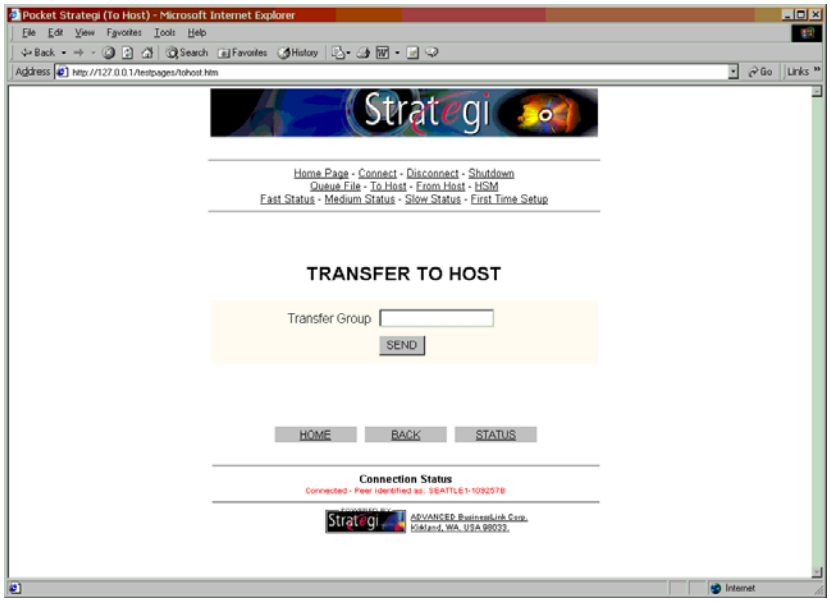
Properties are used to set information about the event for processing the queued file. Each property set will have a corresponding value and be available for retrieval by the handler. Standard properties begin with the '*' character, while user-defined properties must not. Consult the *Strategi Event Services Guide* for a list of standard Pocket Strategi available properties.



To Host

To Host demonstrates how to send queued files to host Strategi by issuing the CONNECT opcode immediately followed by the TOHOST opcode. By issuing the opcodes in this order it ensures a connect request is made before attempting to transfer the files to the host.

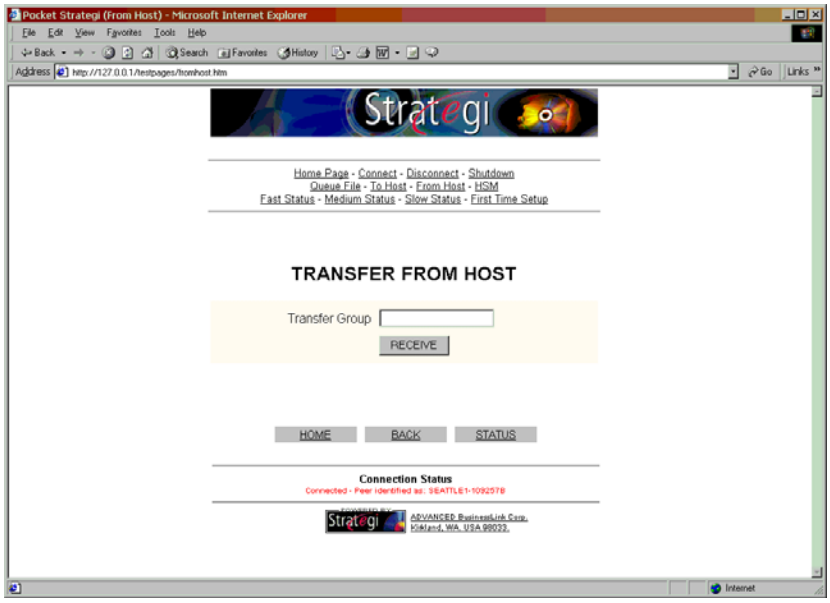
The only parameter required is the Transfer Group name since all other aspects of the file transfer are defined when queuing the file.



From Host

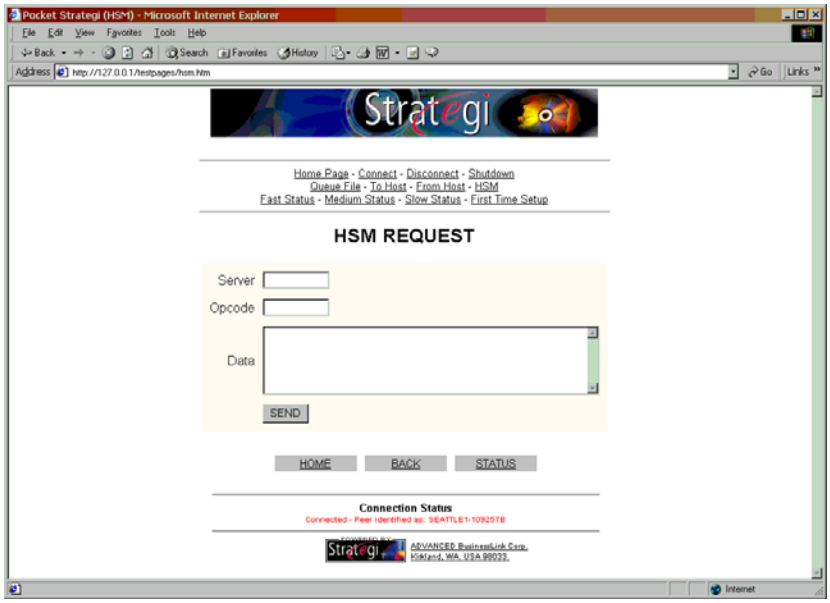
From Host demonstrates how to receive files sent to the Pocket Strategi user by issuing the CONNECT opcode immediately followed by the FROMHOST opcode. By issuing

the opcodes in this order it ensures a connect request is made before attempting to receive the files from the host.. The only required parameter is Transfer Group. It is the responsibility of the Event handler to process the received file as it is handed the event object from Pocket Strategi.



HSM

HSM allows for testing HSM servers by making an HSM request and passing optional request data. For this situation, the CONNECT opcode is first issued, immediately followed by an HSM server request, to ensure a connect request is made since the request could be targeted for an HSM server active on host Strategi.



Status

The Status pages demonstrate how to obtain status information on file transfers and connections by server-side including footer (stdtail.txt) and header (stdhead.txt) files and using a META HTML tag to automatically reload (refresh) the page after one, five or 10 seconds.

File Transfer

Host Strategi and Pocket Strategi Client are capable of sending files to each other. Every file sent must have an Event Code associated with it in order to be considered a Pocket Strategi file transfer. Host Strategi and Pocket Strategi Client both maintain a list of Event Codes they are prepared to accept, but how they maintain the information is different. Host Strategi uses command interfaces for maintaining the information; Pocket Strategi relies upon entries in the eh.dbf file to route the file transfer to the appropriate Event handler via the Event Code.

Pocket Strategi Events are written in Java; host Strategi Events are written in Java or a language native to the platform. The Event Handler receives an Event object (the file sent with that Event Code) and writes the file data to a file on the file system, or wherever the Event Handler author has chosen.

From-Host

Sending files to Pocket Strategi users is accomplished via the Send Strategi File (SNDSGIF) command on the host, with a corresponding Event Handler on the client to process the sent file (Event). SNDSGIF expects a Strategi user number, file location, and several other parameters. In the case of sending to Pocket Strategi Client, it will also expect Transfer Control, comprised of a Transfer Group and Event Code. Since a Pocket Strategi user is defined as a host Strategi user, the differentiating factor to properly route the file is Transfer Control.

A Transfer Group must be assigned to each file sent via SNDSGIF. When Pocket Strategi Client connects and requests to receive files from the host, it must specify the transfer group it wants to receive. It will receive only files sent within the specified transfer

group. As the files are received, Pocket Strategi reads the file properties to determine the Event Code and Transfer Group associated with the sent files and hands the received files to the Event Handler specified by the Event Code. The Transfer Group is a standard Pocket Strategi property available to the Event Handler to retrieve and use, if desired. (See *Strategi Event Services Guide* for a list of standard Pocket Strategi available properties).

SPF Files

SPF files sent from host Strategi to Pocket Strategi Client using SNDSGIF will be sent without data transformation. Specifying the Event Code as *CLIENTUPDATE will ensure Pocket Strategi Client processes the received Event when the FROMHOST opcode is requested for the appropriate Transfer Group (value specified at file sending). For example,

```
SNDSGIF +
  USER(000000002) +
  FROMFMT(*IFS) +
  IFSFILE('/psgiapp/myapp/currentspf/MyApp.spf') +
  TFRCTL(SGIUPDATE *CLIENTUPDATE)
```

DB2/400 Files

DB2/400 files sent from host Strategi to Pocket Strategi Client using SNDSGIF will undergo transformation into a dBase III (DBF) file if sent with a TOFMT parameter value of *XBASE. There must be a Pocket Strategi Event handler configured on the client to handle the file once received. For example,

```
SNDSGIF +
  USER(000000002) +
  FROMFMT(*DB2400) +
  FILE(MYLIB/SOMEFILE) +
  TOFMT(*XBASE) +
  TOFILE('somefile.dbf') +
  TFRCTL(MYXFER MYEVENT)
```

To-Host

Sending files from Pocket Strategi to host Strategi is accomplished in three steps: queuing the file, making a connection to the host, and sending the queued transfer group to the host via the Pocket Strategi *CLIENT server TOHOST opcode. Be sure to use the CLOSE opcode on the file prior to upload to ensure that the end of file marker has been properly written to the file.

Receiving files on host Strategi from a Pocket Strategi file transfer occurs differently than in Pocket Strategi Client because there is no specific request on the host to receive a file transfer for a specific Transfer Group. Transfer Groups are designed for performance reasons for the client, not host Strategi. As host Strategi receives a Pocket Strategi file transfer (Event), it automatically routes the Event to the proper Event Handler for Event processing.

Consult the *Strategi Event Services Guide* for additional details on configuring host Strategi Events.

High Speed Messaging (HSM)

Strategi's High Speed Messaging (HSM) is an infrastructure that provides separation between client interface and server logic. Pocket Strategi includes HSM support.

Consult the *Strategi HSM Programmer's Guide* for additional details regarding HSM infrastructure. Online Java documentation is also available.

Pocket Strategi will attempt HSM processing on the following file types:

.htm	.html	.shtm	.shtml
.hdml	.hdmlc	.wml	.xml
.txt	.text	.rtf	.hsm

HSM Servers

All HSM servers, regardless of the programming language used to build them, will contain the basic program logic to (1) receive a request, (2) process the request, and (3) send a reply. They provide a convenient method for data manipulation.

When writing HSM servers, without regard to whether they are host or client HSM servers, the following Skeleton server code is the starting point for developing new HSM servers.

Skeleton Server Code

Use this HSM Skeleton server code as the basis for building new HSM servers to decrease application development time. Feel free to copy and paste the following lines of code to use in writing your first HSM servers.

```
/* ---------------------------------------------------------
 * This is a minimal sample HSM Server written in Java.
 *
 * The code is the bare minimum needed to illustrate the
 * use of the HSM class and provide a working server.
 *
 * ---------------------------------------------------------*/

import com.businesslink.sgi.api.hsm.*;

public class SkeletonServer
extends HsmServer
{

public SkeletonServer() {
    super("Skeleton HSM Server, Build 1.2.3"); // Give a good description
      for *PING
    }

public void startHsmServer(String svrnam) {
    System.out.println("Skeleton HSM Server");
    }

public void endHsmServer() {
    System.out.println("Skeleton HSM Server Ending");
    }

public boolean processHsmRequest(HsmServerRequest hsr,HsmClientDetails
  clt) {  // return true if still active
    String           opc;

    opc=hsr.getRequestOpcode();
    if(opc.equals("OPCODE1")) {
        System.out.println("Received OPCODE1");
        hsr.setReplyOpcode("DONE");
        hsr.setReplyData("Processed OPCODE1");
        return true;
        }
    else if(opc.equals("OPCODE2")) {
        System.out.println("Received OPCODE2");
```

```
        hsr.setReplyOpcode("DONE");
        hsr.setReplyData("Processed OPCODE2");
        return true;
        }
    else {
        return processHsmRequestDefault(hsr,clt);
        }
    }
} // END OF CLASS
```

Server-to-Server Code

An HSM server is also capable of making requests of other HSM servers, in effect, becoming an HSM client to another HSM server. In the Strategi user community this is commonly referred to as a Server-to-Server call. One such use of server-to-server calls is an HSM server making an HSM client request to the *CLIENT server.

The following example demonstrates how to formulate an HSM client request from another HSM server. The request is to server *CLIENT for opcode NRMPATH with a path relative to Pocket Strategi root of "hsmservers."

```
. . .
if(opc.equals("MYREQUEST")) {
    System.out.println("Received MYREQUEST");

    HsmClient        hc;
    HsmClientRequest hcr;
    hc=new HsmClient();
    hcr=new HsmClientRequest();

    hcr.resetRequest();
    hcr.setRequestOpcode("NRMPATH");
    hcr.setRequestData(1,1024,"hsmservers");

    hc.processHsmRequest("*CLIENT",hcr);
    if(hcr.getReplyOpcode().equals("NRMPATH")) {
        hsr.setReplyOpcode("MYREQUEST");
        hsr.setReplyData(1,9999,hcr.getReplyString(1,9999));
        }
    else {
        hsr.setReplyOpcode("ERROR");
        hsr.setReplyData("Problems getting the system path. Reply was:" +
          hcr.getReplyString(1,9999));
        }
    hc.close();
    return true;
    }
. . .
```

Pocket Strategi Client Opcodes

Pocket Strategi's HSM-enabled webserver contains built-in functions to allow an application to make HSM requests to and receive replies from Pocket Strategi Client. The HSM server name to reference when issuing HSM requests is *CLIENT.

The following section details each supported opcode for the *CLIENT server. Since *CLIENT is a built-in server, it is not necessary to define *CLIENT in the hm.dbf file.

All requests may return the standard HSM errors as well as the replies detailed below. The *ERROR replies have the standard seven character code plus one space plus descriptive text layout, with detail potentially varying from release to release.

Request and reply layouts are listed with the field length followed by a description of the field. For arrays, letters are used for variable numbers. For example, Nx15 is an array of 15-character fields with the number of elements N defined previously, and NxL is an array where the length of entry also varies.

Opcode CONNECT

This initiates a request to connect to the host. Pocket Strategi will attempt to make a connection until a DISCONNECT opcode is issued, or a successful connection to the host is made.

Request Opcode CONNECT

128	Host name
40	User name
40	User password

Reply Opcode CONNECT

*	Descriptive text

Example

```
[server request]
server=*CLIENT
opcode=CONNECT
start_length_field=1,128,HSTNAM
start_length_field=*,40,USRNAM
start_length_field=*,40,USRPWD

[reply]
opcode=CONNECT
start_length_field=*,*,DSCTXT
```

Opcode DISCONNECT

This initiates a request to disconnect from the host. The disconnect always succeeds, returning descriptive text. To wait for the actual disconnect to complete, use QRYCONNECT requests until the status is 00.

Request Opcode DISCONNECT

No Data

Reply Opcode DISCONNECT

*	Descriptive text

Example

```
[server request]
server=*CLIENT
opcode=DISCONNECT

[reply]
opcode=DISCONNECT
start_length_field=1,*,DSCTXT
```

Opcode FROMHOST

This initiates a Pocket Strategi file transfer from the host to the user for a given transfer group.

Request Opcode FROMHOST
15 Transfer group

Reply Opcode FROMHOST
No Data

Example

```
[server request]
server=*CLIENT
opcode=FROMHOST
start_length_field=1,15,"SGIUPDATE"

[reply]
opcode=FROMHOST
```

Opcode GETCONNECT

This gets connection detail information used for connecting to the host.

Request Opcode GETCONNECT
No Data

Reply Opcode GETCONNECT
128 Host name
40 User name
1 Password required flag
9 User number

Example

```
[server request]
server=*CLIENT
opcode=GETCONNECT

[reply]
opcode=GETCONNECT
start_length_field=1,128,HSTNAM
start_length_field=*,40,USRNAM
start_length_field=*,1,PWDRQD
start_length_field=*,9,USRNUM
```

Opcode NRMPATH

This gets the full system path to an object normalized to remove relative path specifications, and protected to be no higher than Pocket Strategi's install path ("../" parts that would take it above Pocket Strategi' root are ignored), given the path relative to Pocket Strategi root.

Request Opcode NRMPATH
1024 Path to normalize

Reply Opcode NRMPATH

9999 Full system path

Example

```
[server request]
server=*CLIENT
opcode=NRMPATH
start_length_field=*,1024,RELPATH

[reply]
opcode=NRMPATH
start_length_field=*,9999,FULLPATH
```

Opcode QRYCONNECT

This gets the connection status.

Request Opcode QRYCONNECT

No Data

Reply Opcode QRYCONNECT

2 Connection status code (00=offline; 01=online; 02=connect in progress; 03=disconnect in progress)

200 Connection status text

Example

```
[server request]
server=*CLIENT
opcode=QRYCONNECT

[reply]
opcode=QRYCONNECT
start_length_field=1,2,CNNSTSCDE
start_length_field=*,200,CNNSTSTXT
```

Opcode QRYEVENTS

This gets the event status for online and offline events. Can be used to wait until event handling is complete before continuing in a process.

Request Opcode QRYEVENTS

1 Online/offline queue (1=online; 2=offline)

Reply Opcode QRYEVENTS

2 Status code (00=inactive, no events pending; 01=active, events being processed; 02=held or ended with events still remaining)

200 Status code text

Example

```
[server request]
server=*CLIENT
opcode=QRYEVENTS
start_length_field=1,1,"1"

[reply]
```

```
opcode=QRYEVENTS
start_length_field=1,2,ONLINESTSCDE
start_length_field=*,200,ONLINESTSTXT

[reply]
opcode=*OTHER
assign_field_value=ONLSTSCDE,"00"
assign_field_value=ONLSTSTXT,*REPLY.OPCODE

[server request]
server=*CLIENT
opcode=QRYEVENTS
start_length_field=1,1,"2"

[reply]
opcode=QRYEVENTS
start_length_field=1,2,OFFLINESTSCDE
start_length_field=*,200,OFFLINESTSTXT

[reply]
opcode=*OTHER
assign_field_value=OFFLINESTSCDE,"00"
assign_field_value=OFFLINESTSTXT,*REPLY.OPCODE
```

Opcode SENDFILE

This queues a file for sending to the host. Before a file can be sent to the host via opcode TOHOST, the file must first be queued via opcode SENDFILE.

Request Opcode SENDFILE

256	File name
15	Transfer group
15	Event code
3.0	Property count
50x15	Property name array
50x128	Property value array

Reply Opcode SENDFILE

10	Transfer group tracking number

Example

```
[server request]
server=*CLIENT
opcode=SENDFILE
start_length_field=*,256,&FILENAME
start_length_field=*,15,&TRANSFERGROUP
start_length_field=*,15,&EVENTCODE
start_length_field=*,3.0,&PRPCNT
start_length_field=*,50x15,&PROPNAME
start_length_field=*,50x128,&PROPVALUE

[reply]
opcode=SENDFILE
start_length_field=*,10.0,&TGTNUMBER
```

Opcode SHUTDOWN

This initiates a request to shutdown Pocket Strategi.

Request Opcode SHUTDOWN
No Data

Reply Opcode SHUTDOWN
No Data

Example

```
[server request]
server=*CLIENT
opcode=SHUTDOWN

[reply]
opcode=SHUTDOWN
```

Opcode TFRSTATUS

This gets the transfer status of files sent to and from the host.

Request Opcode TFRSTATUS
1 Sent or received transfer indicator (1=from host; 2=to host)
1 Clear status flag (0=clear; 1=do not clear)

Reply Opcode TFRSTATUS
2 Status code (00=disconnected; 01=connected; 02=connecting; 03=disconnecting)
200 Status code text

Note on the clear flag: If this is set and the transfer is at final status, the status will be reset to "No Transfer." This should only be set by the thread responsible for tracking the transfer, usually the one that initiated it. Other secondary threads may query the status without clearing to provide a display to the user.

Example

```
[server request]
server=*CLIENT
opcode=TFRSTATUS
start_length_field=1,1,"1"
start_length_field=*,1,"0"

[reply]
opcode=TFRSTATUS
start_length_field=1,2,FMHSTSCDE
start_length_field=*,200,FMHSTSTXT

[reply]
opcode=*OTHER
assign_field_value=FMHSTSCDE,"00"
assign_field_value=FMHSTSTXT,*REPLY.OPCODE
```

Opcode TOHOST

This initiates a Pocket Strategi file transfer to the host for a given transfer group.

Request Opcode TOHOST
15 Transfer group

Reply Opcode TOHOST

No Data

Example

```
[server request]
server=*CLIENT
opcode=TOHOST
start_length_field=*,15,TRANSFERGROUP

[reply]
opcode=TOHOST
```

Pocket Strategi Host Server

Communication with the Pocket Strategi Host Server is available to allow starting and stopping Pocket Strategi using HSM. The server for this is *HSMMGR.

Request Opcode *START

10	Server Name
1	Instance Type (“M”=Multiple; “S”=Single)
128	Class Name
1024	Class Path

Reply Opcode DONE
Reply Opcode *ERROR

Request Opcode *STOP

10	Server Name

Examples

```
-----------------------------------------
[SERVER REQUEST]
SERVER=*HSMMGR
OPCODE=*START
start_length_field=0001, 0010, "SERVERNAME"
start_length_field=0011, 0001, "S"
start_length_field=0012, 0128, "ClassName"
start_length_field=0140, 1024, "classpath

[REPLY]
OPCODE=DONE

[REPLY]
OPCODE=*OTHER

-----------------------------------------
[SERVER REQUEST]
SERVER=*HSMMGR
OPCODE=*START
start_length_field=0001, 0010, "SERVERNAME"
start_length_field=0011, 0001, "M"
start_length_field=0012, 0128, "ClassName"
start_length_field=0140, 1024, "classpath

[REPLY]
OPCODE=DONE
```

```
[REPLY]
OPCODE=*OTHER

----------------------------------------
[SERVER REQUEST]
SERVER=*HSMMGR
OPCODE=*STOP
start_length_field=0001, 0010, "SERVERNAME"

[REPLY]
OPCODE=DONE

[REPLY]
OPCODE=*OTHER
```

HSM Data Servers

Strategi HSM Data Servers are designed to provide convenient access to databases from HSM resource files. The same access is also available from any HSM client program but is generally less convenient than direct database programming.

Strategi provides two implementations of HSM Data Servers, MiniDBMS and JDBCDS. Both are Java HSM servers, and their HSM server request and reply codes and formats are the same so applications can run against either server.

MiniDBMS is intended for single-user client systems running under Pocket Strategi. JDBCDS is intended for multi-user data access on any system that provides an underlying DBMS supporting JDBC SQL access to files. It is expected the data files contain data for all users. Server choice, if any, is dependent upon intended application usage.

Consult the *Strategi HSM Data Servers* manual for details on setup, configuration, file access and opcode layout.

Events

Strategi's Event infrastructure provides for event processing on the host as well as client device. A Strategi Event is described as an exchange between a Strategi Event Client and a Strategi Event Handler, in which properties (keyword/value pairs) and data are sent and received. The exchange follows a strict sequence, outlined as follows:

1. The client sends properties to the handler
2. The client sends data to the handler
3. The handler sends properties to the client
4. The handler sends data to the client

There is no limit on the number of properties or the size of the data exchanged in either direction, except for possible device limitations. Not all Event handlers will be designed to send properties and data in response to the client, but if it does, it must follow the required sequence.

Events are the mechanism used to receive transferred files. There are two types of Event handlers: system-defined and user-defined. A system-defined Event handler manages Events specifically related to the core application environment, such as the

*CLIENTUPDATE Event. A user-defined Event handler processes Events outside the core environment, such as placing files into other directories on the client device.

Java documentation for Events is available from ADVANCED BusinessLink's Manuals website at http://manuals.businesslink.com/.

Consult the *Strategi Event Services Guide* for details on configuring and registering host Event handlers.

Pocket Strategi Event Handlers

Pocket Strategi Event handlers are designed to process a single Strategi event. Pocket Strategi Event handlers are Java classes and must be registered in the eh.dbf file. All user-defined Event handler classes must be stored in the "eventhandlers" directory.

The online Java documentation details the classes and methods used to process an Event; the *Strategi Event Services Guide* details available properties to get and send (set).

Example Code

The following Pocket Strategi Event Handler source code examples demonstrate API usage.

LogProperties
The "LogProperties" Event handler demonstrates getting all properties associated with the Event and outputting the values to the print writer.

```
/* ------------------------------------------------------------
 * LogProperties Event Handler to Demonstrate Getting All
 * Properties Associated with the Event
 * ------------------------------------------------------------ */

import java.io.*;
import java.util.*;
import com.businesslink.sgi.api.*;

public class LogProperties
implements StrategiEventHandler
{
private PrintWriter        prtwtr;  // print writer

public void processEvent(StrategiEvent evt) throws Throwable {
    prtwtr=evt.getTextWriter();
    prtwtr.println("Beginning LogProperties");
    logProp(evt);
    prtwtr.println("Ending LogProperties");
    }

private void logProp(StrategiEvent evt) {
    Properties             prp;    // properties

    prtwtr.println("Reading event properties");
    prp=evt.getProperties();
    prp.list(prtwtr);
    }

} /* END PUBLIC CLASS */
```

ReceiveFile
The "ReceiveFile" Event handler demonstrates Event property retrieval used for building the received file output.

```
/* ------------------------------------------------------------
```

```
 * ReceiveFile Event Handler to Demonstrate Event Property
 * Retrieval Used for Building the Received File Output
 * ------------------------------------------------------------- */

import java.io.*;
import com.businesslink.sgi.api.*;

public class ReceiveFile
implements StrategiEventHandler
{
private PrintWriter        prtwtr;  // print writer

public void processEvent(StrategiEvent evt) throws Throwable {
    prtwtr=evt.getTextWriter();
    prtwtr.println("Event ReceiveFile beginning");
    writeFile(evt);
    prtwtr.println("Event ReceiveFile complete");
    }

private void writeFile(StrategiEvent evt) {
    String                 fileName;
    String                 ifsDir="c:\\temp\\"; // could use a property
    OutputStream           outStream;
    long                   tot;
    File                   outFile,dirFile;

    try{
        dirFile=(new File(ifsDir));
        if(!dirFile.exists()) {dirFile.mkdirs();}

        // build file name based upon standard Event properties
        fileName=(evt.getProperty("*TRACKINGNUMBER")).trim() + "_" +
          (evt.getProperty("*FILENAME")).trim();

        outFile=new File(dirFile,fileName);
        prtwtr.println("Writing data to :" + outFile.getName() + ":");

        outStream=new FileOutputStream(outFile);
        tot=evt.writeDataTo(outStream);
        outStream.close();

        prtwtr.println("Wrote event data. " + tot + " bytes");
        }
    catch (IOException e) {
        prtwtr.println("Exception thrown");
        e.printStackTrace(prtwtr);
        }
    }

} /* END PUBLIC CLASS */
```

Administration

Pocket Strategi administration is performed on the host Strategi system and is generally categorized into six major areas: Licensing, Host Configuration, New User Configuration, Pocket Strategi Client Updates, Application Packaging, and Application Deployment.

Licensing

Pocket Strategi is a licensed component of Strategi and requires two licenses, one for Pocket Strategi Host, the other for Pocket Strategi Users. Contact your Sales Representative for details on pricing.

Both Pocket Strategi licenses are set using the Strategi Set License Information (SETSGILIC) command. License setting instructions are included at the time of license

key issuance. Help text is available for SETSGILIC using standard IBM AS/400 help text prompting.

Host Configuration

The one Strategi Value that requires modification from its default value is PSGIADDRESS. PSGIADDRESS is initially set to *NONE for the TCP/IP Address value and 43840 for the TCP/IP Base Port. TCP/IP Address must be changed to an allowable value. (Consult the *Strategi Administrator's Guide* for details on properly configuring PSGIADDRESS.)

New User Configuration

A Pocket Strategi user is a Strategi user who is allowed access to the Pocket Strategi product; however, a Strategi user can exist without having access to Pocket Strategi.

A Pocket Strategi user is configured in host Strategi using the xxxSGIUSR command set to create, change and delete Strategi user records. (Consult the *Strategi Administrator's Guide* for details on properly configuring Strategi user records.)

Step 1 – Configure the Strategi user to allow access to the Pocket Strategi product.
Using the xxxSGIUSR commands, create the Strategi user, or change an existing Strategi user, and set the Allow Pocket Strategi parameter to *YES.

Step 2 – Allow access to website zone PSTRATEGI.
Using the xxxSGIZNA commands, add zone authority for the Strategi user to allow *READ authority to the PSTRATEGI website zone on the RESOURCES website. (Consult the *Strategi Administrator's Guide* for details on properly granting website zone access.)

Once the user is configured to allow Pocket Strategi access, the user is ready to download Pocket Strategi Client and initiate the first-time connection. Any files sent to the user specifically related to Pocket Strategi, such as the client application or application updates, will be downloaded to the user.

Pocket Strategi Client Updates

The *pktsgi* directory is located in the host Strategi IFS root directory. It contains the Pocket Strategi Client and Pocket Strategi Client Update SPF files.

```
/strategi/pktsgi/pStrategi.spf
/strategi/pktsgi/pStrategi-Update.spf
```

The Pocket Strategi Client, pStrategi.spf, file is reserved for future use.

The Pocket Strategi Client Update, pStrategi-Update.spf, file is for existing Pocket Strategi users who need their Pocket Strategi Client updated to the latest version.

Pocket Strategi Client has a built-in Processing Event Code *CLIENTUPDATE to use when updating the Pocket Strategi environment. It writes the SPF file to the root of the Pocket Strategi directory as pStrategiUpdate.spf. In order to apply the update, Pocket Strategi must be shut down and restarted. Each time Pocket Strategi is started it looks for a pStrategiUpdate.spf file, and if found, processes the file.

The Transfer Group associated with the update must be requested in the FROMHOST *CLIENT opcode request in order to receive the file transfer.

*CLIENTUPDATE is valid for updating Pocket Strategi Client or an application update packaged as an SPF file. Create custom Event Handlers to process received files not packaged or prepared for *CLIENTUPDATE use.

To send a Pocket Strategi Client update to a user, execute SNDSGIF passing the required parameters. For example,

```
SNDSGIF +
  USER(000000002) +
  FROMFMT(*IFS) +
  IFSFILE('/strategi/pktsgi/pStrategi-Update.spf') +
  TFRCTL(SGIUPDATE *CLIENTUPDATE)
```

Once the Pocket Strategi user makes a connection to the host and requests a FROMHOST transfer specifying Transfer Group SGIUPDATE, the file is received and prepared for updating Pocket Strategi Client.

Pocket Strategi Client Update Results

On the Windows client, Pocket Strategi is updated using the pStrategi.jar file, which is a very simple program used to load updates and perform other simple processes. The process is as follows:

1. After download, the pStrategi-Update.spf file exists in the root of Pocket Strategi installation
2. On the next startup of pSGI:
   a. New files are created in the pSGI root: pStrategi.jar.new, pStrategiClient.jar.new, pStrategiUpdate.jar.new
   b. The existing pStrategiClient.jar and pStrategiUpdate.jar files have .old appended to the file name (for archival or rollback purposes)
   c. The pStrategiClient.jar.new and pStrategiUpdate.jar.new files are renamed to remove the .jar in the file name
   d. The pStrategi.jar file remains intact, and the pStrategi.jar.new file is left in the directory

There is no expectation that the pStrategi.jar file will require updating. If the file does require replacement, it can be accomplished through simply renaming the existing pStrategi.jar and pStrategi.jar.new files to pStrategi.jar.old and pStrategi.jar respectively.

It is normal and correct for pStrategi.jar.new to remain after a pStrategi upgrade on Windows systems, where active jarfiles cannot be renamed or deleted, so whatever takes the initial upgrade steps cannot itself be updated. It is unlikely that this jarfile will ever have to be changed, since its role is simply to allow replacement of the main jarfiles. But it is issued as a safety net so a manual rename process can be used if ever needed (and of course it updates automatically on Unix-based systems).

Note that if the pStrategiClient.jar had been placed on the classpath in the Windows System Properties, Environment Variables, the pStrategiClient.jar.new will not be applied.

Application Packaging

Strategi provides a host-based application packaging mechanism for packaging Pocket Strategi applications. Once the client portion of the application is developed, including all Event Handlers, HSM Servers, database files, etc., the next step is to move it onto the host for packaging and deployment.

Packaging is done with a Java class provided specifically for this purpose. The class is

```
com.businesslink.sgi.rmt.dist.PSGIDistribution
```

and is part of the StrategiAS400.jar file. It requires two parameters: a fully qualified output file and source input.

The output file will consist of the complete IFS path and filename, with a file extension of SPF. For example,

```
/psgiapp/myapp/currentspf/MyApp.spf
```

The inputs can come from anywhere, but trailing directory structure must be the same as under the Pocket Strategi structure. In practice, the trailing directory should contain a replica of the Pocket Strategi environment for the objects affected by the application. This normally includes the application, database, eventhandlers, hsmservers, and website directories, including any subdirectories located therein. Also affected may be the eh.dbf, hm.dbf and tg.dbf files. Each Event Handler, HSM Server and Transfer Group added to the application environment will require an entry into the appropriate Pocket Strategi Client core database file.

It is important to package only objects necessary for the application. For example, when packaging a Pocket Strategi-based application containing HSM servers, it is not necessary to include the entire database directory, or even the entire eh.dbf. Instead, create a modified version containing only records to add or change within eh.dbf. Name the file upd.eh.dbf and place that into the database directory for packaging. When the application is unpackaged on the client, Pocket Strategi will update eh.dbf with the contents of upd.eh.dbf, thus guaranteeing the file is updated only with the newly processed records.

The resulting SPF file uses standard Java ZIP compression, but adds Pocket Strategi-specific headers to simplify fast, recoverable transmission, making it a non-standard format.

Files are stored with a path as well as filename, and on restore, that path is treated as being relative to the Pocket Strategi root directory.

To distinguish between the source input path and the path relative to the Pocket Strategi root directory, a '/./' in the input path marks the start of the relative path to be stored (include) and restore to. If omitted, it is assumed at the end of the input path. In other words, only the directories recursed into are stored, not the base input path.

In order to exclude some entries from inclusion into the packaged file, a *EXCLUDE entry introduces stored paths to be excluded from recursion.

Packaging Example

The following example demonstrates how to use the PSGIDistribution class from a Control Language (CL) program on the AS/400 host Strategi system.

Note: The program source code listed here relating to eventhandlers, hsmservers and website is used for illustration purposes only, and is redundant given the root and all subsequent directories of "install" are included (i.e., '/psgiapp/myapp/install/.') for packaging. The purpose in including those entries is to demonstrate how to specifically include one or more files for packaging without including the entire directory they reside in.

```
/* ------------------------------------------------------------------- */
/* Directory containing complete client application:                   */
/*  '/psgiapp/myapp/install'                                           */
/*  which will contain the following sub-directories:                  */
/*    '/application'                                                   */
/*    '/database'                                                      */
/*    '/eventhandlers'                                                 */
/*    '/hsmservers'                                                    */
/*    '/website'                                                       */
/*  and '/website' will contain additional subdirectories.             */
/* Fully qualified outputFile:                                         */
/*  '/psgiapp/myapp/currentspf/MyApp.spf'                              */
/* ------------------------------------------------------------------- */

PGM

DCL &SPFINS     *CHAR  128 '/psgiapp/myapp/currentspf/MyApp.spf'

ADDSGICLSP
RUNJVA CLASS('com.businesslink.sgi.rmt.dist.PSGIDistribution') +
 PARM( +
  &SPFINS +
  '/psgiapp/myapp/install/.' +
  '/psgiapp/myapp/install/./database/upd.eh.dbf' +
  '/psgiapp/myapp/install/./database/upd.hm.dbf' +
  '/psgiapp/myapp/install/./eventhandlers' +
  '/psgiapp/myapp/install/./hsmservers' +
  '/psgiapp/myapp/install/./website/default/live/homepage.htm' +
  '/psgiapp/myapp/install/./website/default/live/homepage.hsm' +
  '/psgiapp/myapp/install/./website/default/live/myApp' +
  '*EXCLUDE' +
  'database' +
  ) +
 OPTIMIZE(30)
CHGOWN OBJ(&SPFINS) NEWOWN(SGIOBJOWN)

ENDPGM
/* ------------------------------------------------------------------- */
```

The source inputs mean the initial list of paths, ahead of *EXCLUDE, are includes, using the '/./' logic described. The database directory is excluded, except for the two explicitly included files, upd.eh.dbf and upd.hm.dbf.

Include
`/psgiapp/myapp/install`
and subdirectories with subdirectories restored relative to Pocket Strategi root, including explicitly files
`/psgiapp/myapp/install/./database/upd.eh.dbf`
`/psgiapp/myapp/install/./database/upd.hm.dbf`
despite the database subtree being excluded.

Include
`/psgiapp/myapp/install/./eventhandlers`
with subdirectories restored relative to eventhandlers.

Include
`/psgiapp/myapp/install/./hsmservers`
with subdirectories restored relative to hsmservers.

Include explicitly files
`/psgiapp/myapp/install/./website/default/live/homepage.htm`
`/psgiapp/myapp/install/./website/default/live/homepage.hsm`

Include
`/psgiapp/myapp/install/./website/default/live/myApp`
with subdirectories restored relative to myApp.

Exclude
```
  database
```
but do include the explicitly named dbf files.

Include implicitly
```
  application
```
with subdirectories restored relative to application.

Database File Updates

There are two primary ways to update Pocket Strategi database files: replace the entire file or update the existing file with new or updated records.

The first method involves packaging the entire database file, such as hm.dbf, complete with all necessary entries to ensure existing plus new entries are included. This method has the potential risk of not all required entries being present in the new file if modified.

The second method involves packaging the database file with changes only, not the entire database record including old as well as new entries. By packaging the database file with changed records only, there is less potential for changes to adversely affect existing functionality.

To utilize the second method, build the modified dbf file(s) and name them the same as the file to update, except pre-appended with "upd.". Then package the "upd.xx.dbf" database file(s) for distribution.

During the client update process, the database directory is searched for files that begin "upd.", and if found, the file is merged to the filename that matches when the "upd." part is removed (e.g., "upd.cv.dbf" will be merged to "cv.dbf"). Once applied, upd.xx.dbf is removed from the database directory. This allows HSM Servers (hm.dbf), Event Handlers (eh.dbf) and select CV values (cv.dbf) to be updated/added without distributing the entire file.

Application Deployment

Having packaged a Pocket Strategi application by using the PSGIDistribution class, deploying the application to the Pocket Strategi users involves using the SNDSGIF command to send the SPF file to the user(s). For example,

```
SNDSGIF +
  USER(000000002) +
  FROMFMT(*IFS) +
  IFSFILE('/psgiapp/myapp/currentspf/MyApp.spf') +
  TFRCTL(SGIUPDATE *CLIENTUPDATE)
```

will send the application to user 000000002 specifying Transfer Group SGIUPDATE.

Using Strategi Groups is a convenient method for sending application updates to a select group of Pocket Strategi users. By including the desired Pocket Strategi users in the same group, the SNDSGIF command only need be executed once to deploy to the entire application base.

The Pocket Strategi user receives the application or application update once they connect to the host and issue the FROMHOST opcode for the Transfer Group containing their application SPF file.

Index